HEP ML/Optimization Go Quantum – QuantISED Pilot

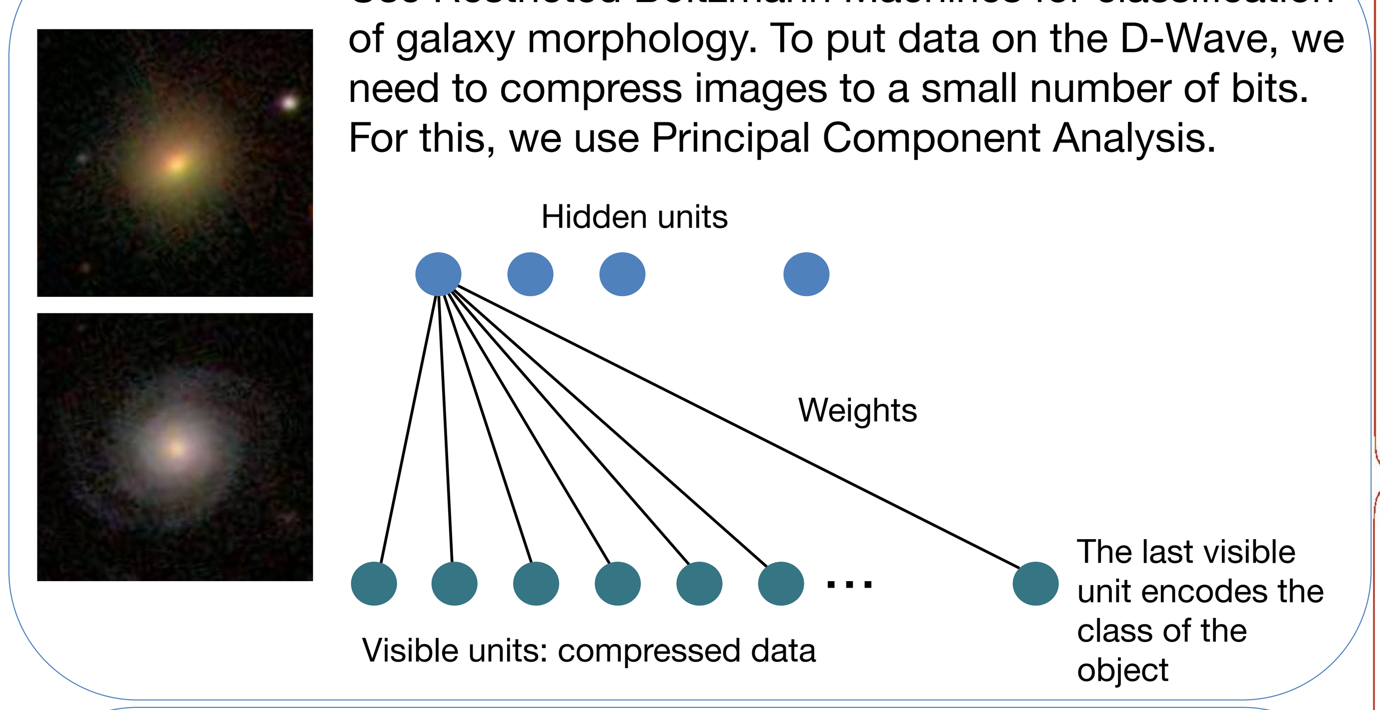
Steven Adachi[3], João Caldeira[1], Andrea Delgado[2], Kathleen Hamilton[2], Travis Humble[2], Joshua Job[3], James Kowalkowski[1], Stefan Leichenauer[5], Alex McCaskey[2], Stephen Mrenna[1], Brian Nord[1], Gabriel Perdue[1], Evan Peters[1,4], Aristeidis Tsaris[1]
[1]Fermi National Accelerator Laboratory, [2]Oak Ridge National Laboratory, [3]Lockheed Martin, [4]University of Waterloo, [5]Alphabet's X

FERMILAB-POSTER-20-042-SCD-QIS

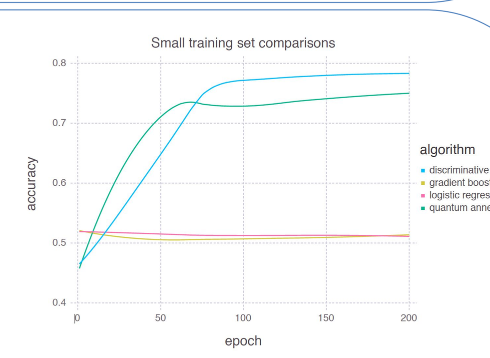
Quantum annealers for classification

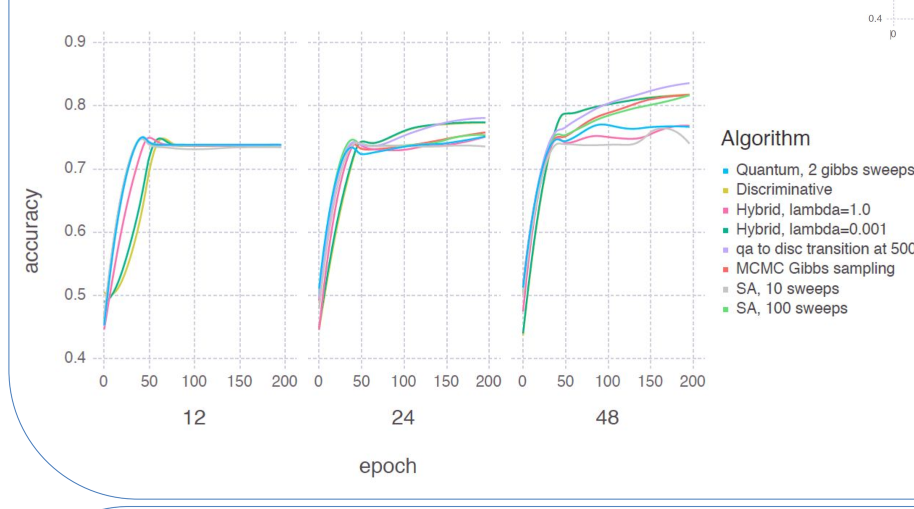
arXiv: 1911.06259

Use Restricted Boltzmann Machines for classification of galaxy morphology. To put data on the D-Wave, we need to compress images to a small number of bits. For this, we use Principal Component Analysis.



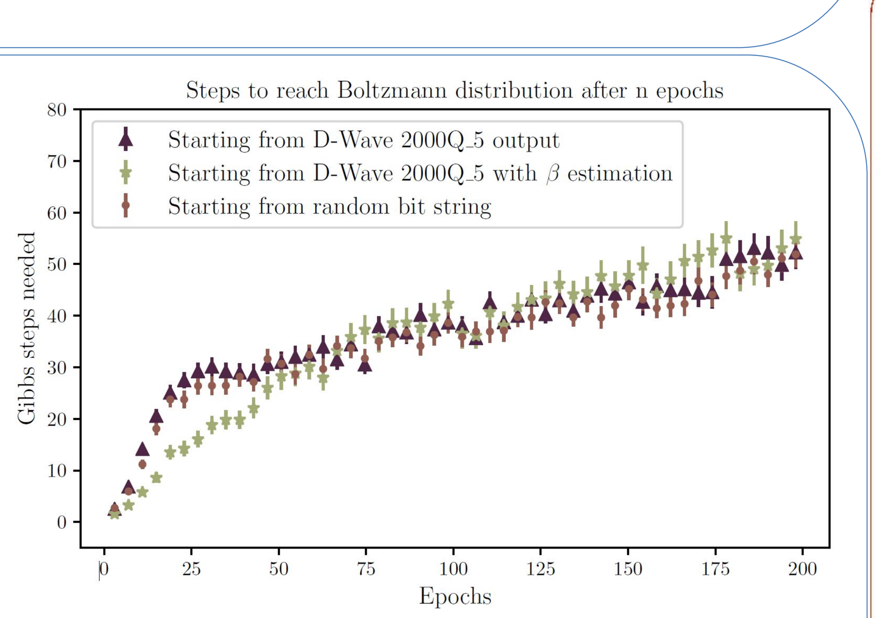
Below, we show the accuracy achieved by RBMs of different size (same number of visible and hidden units, indicated below the graph). Quantum-trained RBMs are competitive at early stages, but dominated by classical RBMs later.





Above, we see RBM beat other classifiers on small training sets (250 examples), and the quantum RBM is the most accurate at early training stages. This test was done with 48x48 RBM.

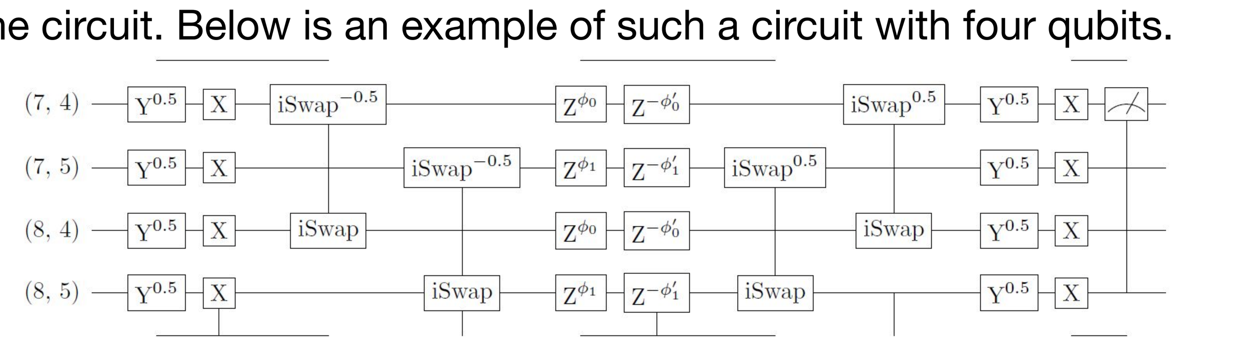
The hard task in training an RBM is sampling from a Boltzmann distribution. We can test how far samples are by using them to seed Gibbs sampling and sampling until Kolmogorov-Smirnov test gives $p > 0.05$.



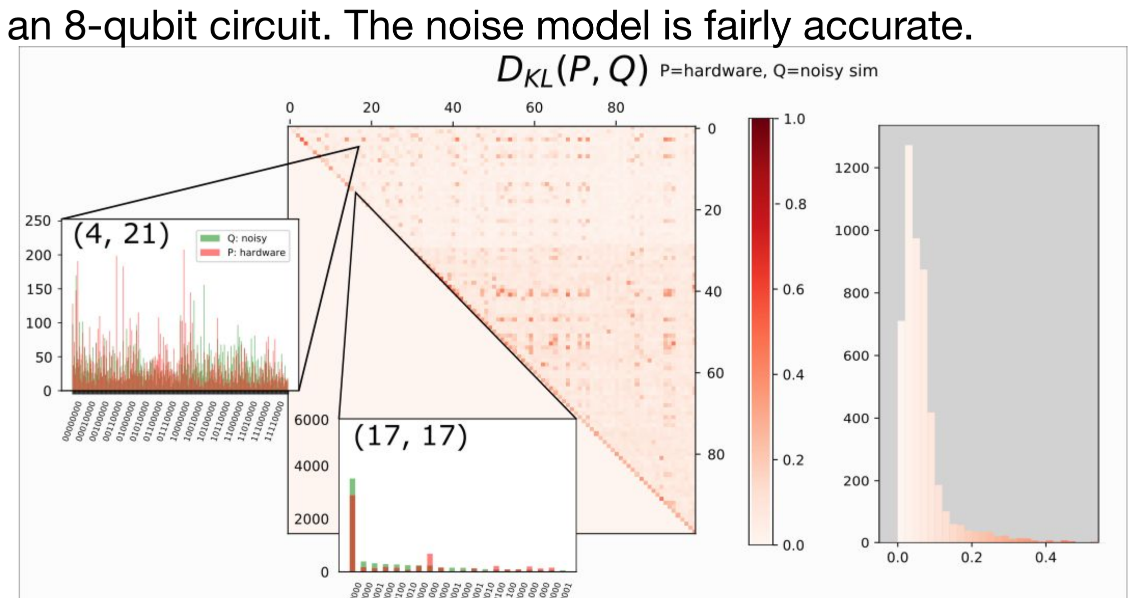
We see using the D-Wave with temperature estimation provides an advantage in early training stages, but that is reduced as couplings increase in later stages.

Quantum gate models for classification

We use Google's Sycamore (below, right) to implement a quantum circuit that maps the data into a high-dim Hilbert space, which we use as a kernel for an SVM. The data determines the rotations in the circuit. Below is an example of such a circuit with four qubits.



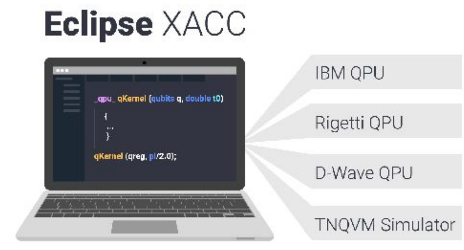
The plots show the KL divergence between the state distribution obtained on the hardware and that obtained on a noisy simulation for an 8-qubit circuit. The noise model is fairly accurate.



Software and Workflows

- Key element of deliverable: workflow and framework software.
  - **How do we make quantum resources available to other HEP scientists and how do we lower the barriers to productivity?**
- XACC (eXtreme-scale ACCelerator)
  - is under development at ORNL
  - abstracts away much of detail of different quantum hardware platforms
  - bundles algorithms with APIs tuned to solving scientific problems.
  - Useful platform for dealing with many hardware issues right now, and HEP work will improve the portfolio of applications for ML and binary CSPs.

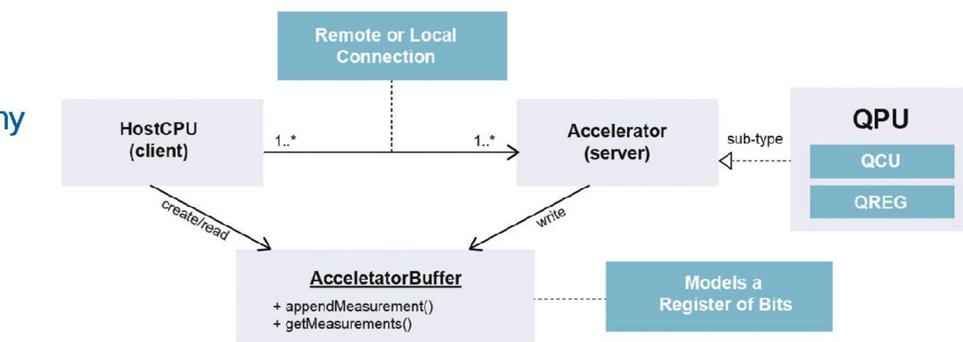
Eclipse XACC



https://github.com/eclipse/xacc

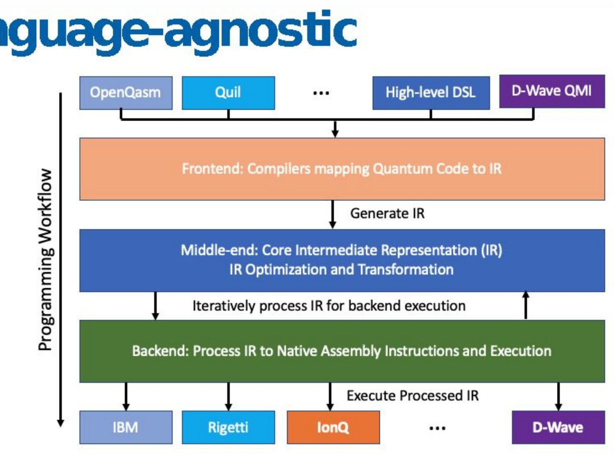
https://www.sciencedirect.com/science/article/pii/S2352711018300700

- **XACC**
  - enables users to write algorithms once for many backends.
  - manages variety of quantum accelerator workflows.



XACC is Language-agnostic

- Simple to use the same algorithm with a variety of backends
- Packages common problems to abstract details of specific quantum frameworks
- Users modify Python code with *decorators* and write generic circuit codes inside the Python shell.



XACC is High-level

- Makes it simple to run common algorithms and applications with very little work.
- Data ingestion is simple and flexible
- Details of hybrid quantum-classical programming are abstracted away.
- Easy to extend when users want to run new algorithms.

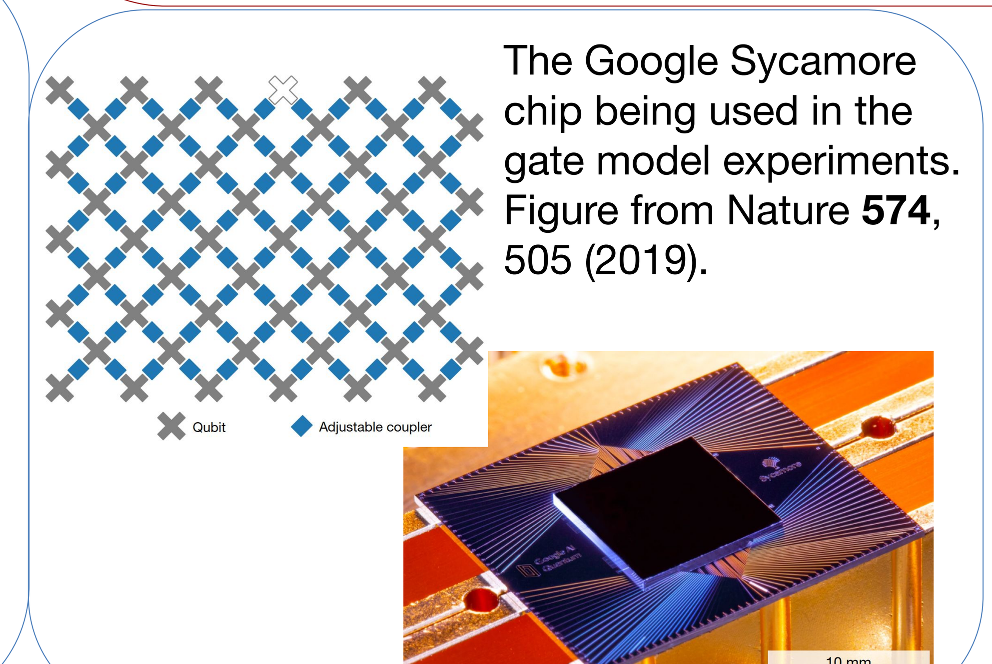
```python
import xacc

# Create the RBM Classification algorithm
algo = xacc.getAlgorithm('rbm-classification',
            {
            'nv':64,
            'nh':64,
            'train-file':'sg_train_64bits.npy',
            'expectation-strategy':'quantum-annealing',
            'backend':'dwave:DW_2000Q_5',
            'shots':100,
            })

qbits = xacc.qalloc()
algo.execute(qbits)

# get the trained RBM weights
# for further use and post-processing
w = qbits['w']
bv = qbits['bv']
bh = qbits['bh']
```

Prototype of a complete *RBM* classifier.

XACC and Hybrid Variational Algorithms

- Algorithm Interface
  - General protocol / data structure for describing hybrid workflows
  - Initialize with problem-specific data
  - Execute, persist results to buffer
- Hardware Agnostic
  - easily swap between gate model QCs from IBM and Rigetti
- Efficient observable definition

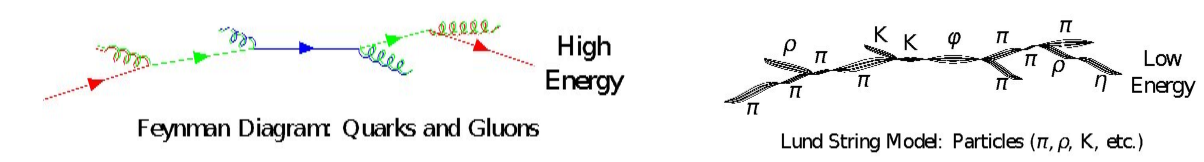
```cpp
#include "xacc.hpp"
#include "xacc_observable.hpp"

int main(int argc, char **argv) {
  xacc::Initialize(argc, argv);

  // Get reference to the Accelerator
  // specified by --accelerator argument
  auto accelerator = xacc::getAccelerator();

  // Create the N=2 deuteron Hamiltonian
  auto H_N_2 = xacc::quantum::getObservable(
      "pauli", std::string("5.907 - 2.1433 X0X1 "
                           "- 2.1433 Y0Y1"
                           "+ .21829 Z0 - 6.125 Z1"));

  auto optimizer = xacc::getOptimizer("nlopt",
                           {std::make_pair("initial-parameters", {.5})});

  // JIT map Quil QASM Ansatz to IR
  xacc::qasm(R"(
 .compiler xasm
 .circuit deuteron_ansatz
 .parameters theta
 .qbit q
 X(q[0]);
 Ry(q[1], theta);
 CNOT(q[1],q[0]);
 )");
 auto ansatz = xacc::getCompiled("deuteron_ansatz");

 // Get the VQE Algorithm and initialize it
 auto vqe = xacc::getAlgorithm("vqe");
 vqe->initialize({std::make_pair("ansatz", ansatz),
               std::make_pair("observable", H_N_2),
               std::make_pair("accelerator", accelerator),
               std::make_pair("optimizer", optimizer)});

 // Allocate some qubits and execute
 auto buffer = xacc::qalloc(2);
 vqe->execute(buffer);

 auto ground_energy = (*buffer)["opt-val"].as<double>();
 auto params = (*buffer)["opt-params"].as<std::vector<double>>();
}
```



The Google Sycamore chip being used in the gate model experiments. Figure from Nature **574**, 505 (2019).

Quantum optimization for LHC physics

- Employ quantum computing to estimate systematics due to Color Recombination models
  - Solving an optimization problem of phenomenological interest
  - Formulate the energy minimization as a binary constraint satisfaction problem
  - Solve for realistic partonic configurations aiming to find a global minimum
- Minimize the color energy subject to constraints.
  - Compare quantum computing results with best-known classical solutions
  - Evaluating the impact on measurements such as top quark mass.



HEP color reconnections in LHC

- Confinement of colored partons into colorless particles may be described with the Lund string model.
- For a given set of quarks and antiquarks, find the arrangement of gluons that minimizes the λ measure

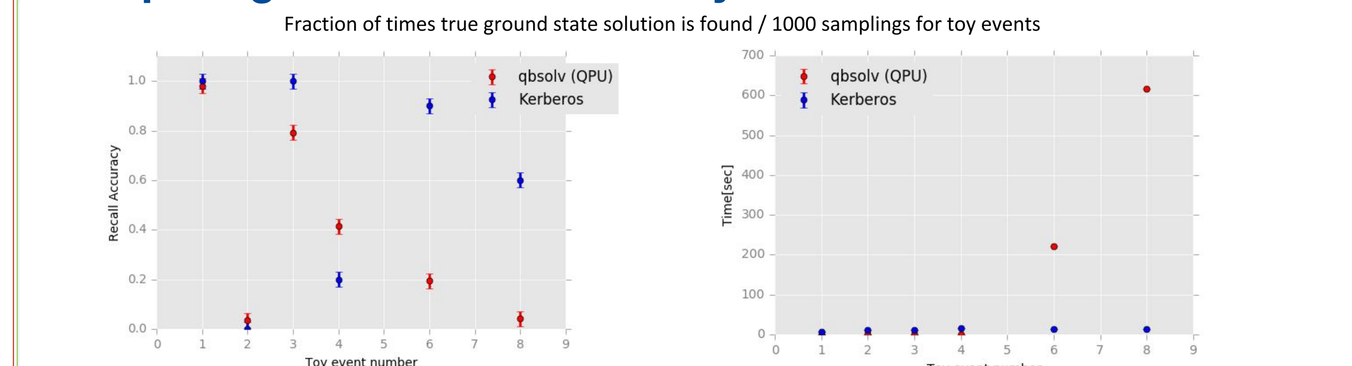
$$\lambda = \sum_{\substack{i=1\\ j=i+1}}^{n+1} ln\frac{m_{ij}^2}{m_0^2} = \prod_{\substack{i=1\\ j=i+1}}^{n+1} ln\frac{m_{ij}^2}{m_0^2}$$

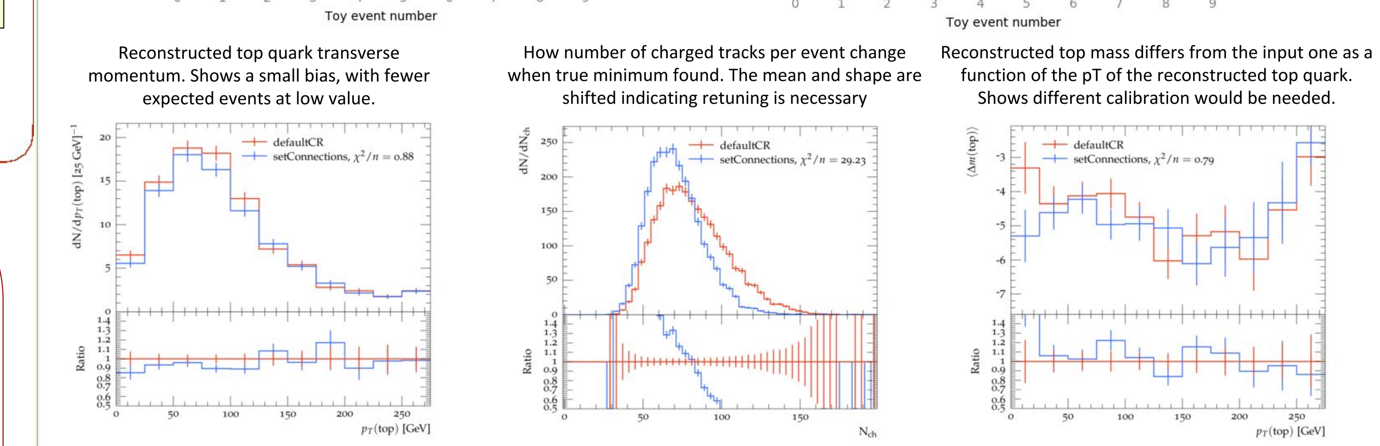
Problem representation using graph theory

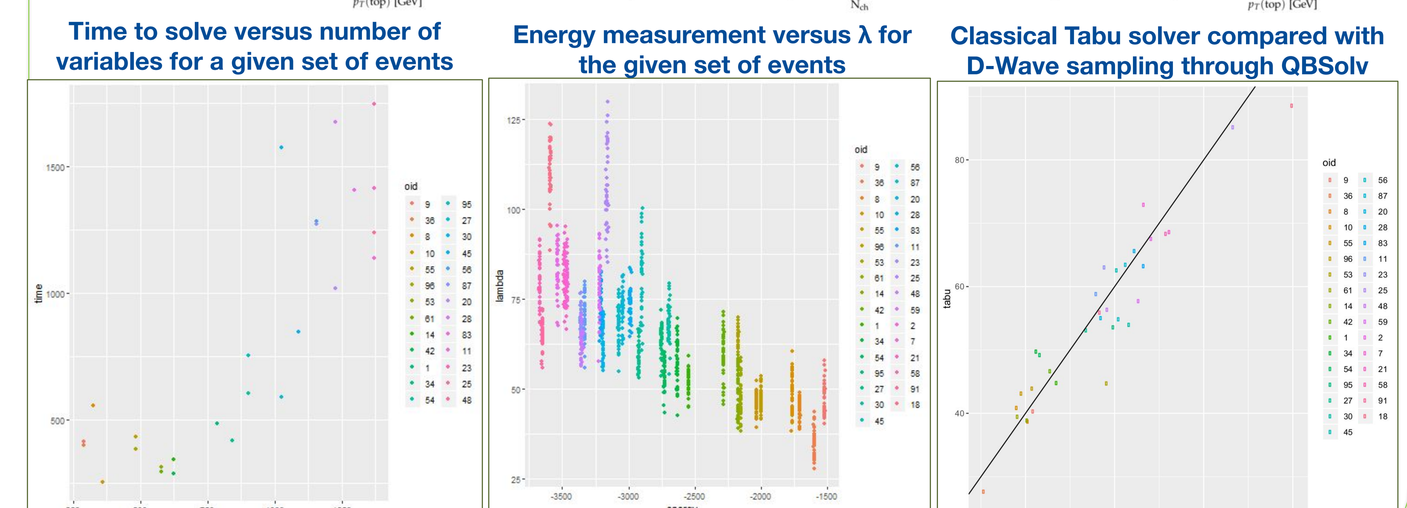
- (1) Particles are embedded into graph G(V;E)
  - Nodes are particles and edges interactions: Quarks are origin nodes O, Gluons are intermediate nodes I, Antiquarks are destination nodes D.
  - Constraints using Hamiltonian path:
    - Every node is in only one position
    - A quark must be in the first position
    - An antiquark must be in the last position
    - Antiquarks are followed by quarks
  - The distance between two nodes is m_{ij} in the λ measure
  - Ising Model variation of the Traveling Salesman Problem
- (2) Connections between particles are nodes, edges form paths
  - Quarks and antiquarks must have one connection
  - Gluons must have two connections
  - Goal is to form one optimal path utilizing all nodes
  - Start with simplest formulation and incrementally add constraints to remove loops or "subtours"

$$H = A\sum_{i\in\mathcal{N}}\Big(1-\sum_{r\in\mathcal{S}}x_{i,r}\Big)^2 + A\sum_{r\in\mathcal{S}}\Big(1-\sum_{i\in\mathcal{N}}x_{i,r}\Big)^2 + A\Big(1-\sum_{k\in\mathcal{O}}x_{k,1}\Big)^2 + A\Big(1-\sum_{l\in\mathcal{D}}x_{l,s}\Big)^2 + A\sum_{r=2}^{s-1}\Big(\sum_{l\in\mathcal{D}}x_{l,r}-\sum_{k\in\mathcal{O}}x_{k,r+1}\Big)^2 + B\sum_{(i,j)\in\mathcal{A}}\lambda_{ij}\sum_{r=1}^{s-1}x_{i,r}x_{j,r+1} - B\sum_{(k,l)\in O\times D}\lambda_{ij}\sum_{r=1}^{s-1}x_{l,r}x_{k,r+1}$$

Comparing solvers and accuracy







Status

- Physics test events generated can be generated on demand
- Classical solver and QUBO formulations complete,
  - Includes functions for mapping events into Hamiltonians
- Gate-based solution using QAOA working for very small events
- Can already run on some platforms
  - AMPL/CPLEX
  - D-Wave using QBSolve with Kerberos, tabu and sampler solvers

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. This work is supported in part by the U.S. Department of Energy, Office of Science, High-Energy Physics Program Office.

IQC Institute for Quantum Computing · IBM QISKit · Cirq · D-Wave The Quantum Computing Company™

Oak Ridge National Laboratory · Fermi National Accelerator Laboratory · Fermilab · U.S. Department of Energy

This work was supported by the DOE Office of Science, Quantum Information Science Enabled Discovery (QuantISED) For High Energy Physics program.